Reichelt, Dirk ; Gmilkowsky, Peter ; Linser, Sebastian :

A study of an iterated local search on the reliable communication networks design problem

Zuerst erschienen in:
Applications of evolutionary computing. - Berlin [u.a.] : Springer, 2005
S. 156-165
(Lecture Notes in Computer Science ; 3449)

A Study of an Iterated Local Search on the Reliable Communication Networks Design Problem

Dirk Reichelt, Peter Gmilkowsky, and Sebastian Linser

Institute of Information Systems, Ilmenau Technical University,
Helmholtzplatz 3, P.O. Box 100565, 98684 Ilmenau, Germany
{Dirk.Reichelt, Peter.Gmilkowsky}@tu-ilmenau.de,
Sebastian.Linser@stud.tu-ilmenau.de

Abstract. The reliability of network topologies is an important key issue for business success. This paper investigates the reliable communication network design problem using an iterated local search (ILS) method. This paper demonstrates how the concepts of local search (LS) and iterated local search can be applied to this design problem. A new neighborhood move that finds cheaper networks without violating the reliability constraint is proposed. Empirical results show that the ILS method is more efficient than a genetic algorithm.

1 Introduction

For many network and internet based IT-applications the error-free operation of the underlying network topology is a key issue for business success. Also, the ongoing integration of IT-systems along the value chain requires high-speed communication networks with low failure probability. Therefore, the availability of communication network topologies is an important factor of design. During the designing process, the designer tries to balance the investments made in the network with the services and benefits provided to its users. One important service measurement of network topology is its all-terminal reliability. This is defined as the probability that all nodes in the network will remain connected, given the probability of success/failure for each node and link in the network [1]. The network design problem dealt with in this paper focuses on choosing those links from a given set of communication links, which minimize the network costs under a given network reliability constraint. The design problem itself, and the calculation of network reliability have been proven as a NP-hard problem[2, 3]. In the past, metaheuristics were successfully applied to the network design process [4, 5, 6, 7, 8].

It is known that the calculation of the all-terminal reliability is the most time-sensitive part of the evaluation of the problems solution. Most of the existing heuristics for this problem require a considerable computational effort in order to evaluate several solutions. For example, population based metaheuristics such

F. Rothlauf et al. (Eds.): EvoWorkshops 2005, LNCS 3449, pp. 156–165, 2005.

as genetic algorithms (GA) proposed in [4, 9, 10, 8] perform a high number of reliability evaluations in each generation. In this paper, we investigate a random restart local search (RRLS) and an iterated local search (ILS). Both methods need significantly fewer fitness/reliability evaluations during the optimization process. For the neighborhood search we define a new 1-by-2-move which minimizes the total network costs with respect to the reliability constraint in each step. With the application of this local search strategy in an ILS, we are able to overcome local optima found by a local search and converge into global optimal solutions. In the empirical results presented, the RRLS and the ILS are compared to a GA using a repair heuristic. We show that an ILS finds optimal solutions with less computational effort when compared to the GA approach.

2 Problem Definition

The work presented here investigates the reliable communication network design (RCND) problem. The challenge is to generate network topologies that satisfy a given reliability measurement while minimizing network costs. The design problem has been proven as NP-hard [2]. Several papers have already been published about this problem and others like it. Dengiz et al. [9] propose a GA using a penalty function to incorporate the reliability constraint into the fitness function. Baran and Laufer [11] build upon the work of Dengiz et al. in order to treat bigger problem situations by using a parallel GA. In [6], Dengiz and Alabap introduce a simulated annealing algorithm. In [8], Reichelt et al. propose a genetic algorithm using a repair heuristic. Baran et al. [12] investigate topology design by a GA with multiple objectives.

In this paper the communication network N is modeled as an undirected simple graph $G(E, V)$, where E is the set of edges and V the set of vertices. Each element of the graph (edge or vertex) represents a link or node in the network. It is assumed that the location of each node is given, setup costs for network nodes are not considered and that, for each possible network link l_{ij} between node i and j, cost c_{ij} and reliability $r(l)$ are known. We do not consider repair of failed edges. It is proposed that nodes are perfectly reliable, and edges are either in an operational or failed state. The failures of the edges are statistically independent with known failure probabilities. The reliability of edge e_{ij} in G is $r(e_{ij})$. A network N as a solution for the problem is represented by subgraph $G_N(E_N, V)$ with $E_N \subset E$. The objective function may be stated as:

$$C(N) = \sum_{i=1}^{|V|} \sum_{j=i+1}^{|V|-1} c_{ij} x_{ij} \rightarrow min$$
$$\text{subject to: } R_{All}(G) \geq R_0 \tag{1}$$

where $C(N)$ is the total cost for the network topology and c_{ij} is the cost for a network link between node i and node j. The variable $x_{ij} \in \{0, 1\}$ indicates whether edge e_{ij} from G representing the network link l_{ij} exists in G_N.

For the reliability measurement we use the all-terminal reliability R_{All}. To determine the all-terminal reliability $R_{All}(G_N)$ we consider a set of states $St = (St_1, \ldots, St_n)$ of the graph G_N. Each state St_i represents a subgraph G_{N_i} of G_N when z edges in G_N fail. A state $St_i \in St$ is operational if G_{N_i} is connected. We define $\Phi(St_i)$=1 if St_i is operational, otherwise $\Phi(St_i)$=0. $Pr(St_i)$ is the probability for state St_i. The all-terminal reliability is:

$$R_{All}(G) = \sum_{St_i \in St} \Phi(St_i) \cdot Pr(St_i) \tag{2}$$

The constraint for $R_{All}(G)$ determines the minimum reliability requirement R_0 for G_N. The calculation of the all-terminal reliability has been proven as NP-hard [3]. For calculation of network reliability the literature proposes exact algorithms [1, 13] for networks with few edges, and Monte Carlo based estimation [14] for large network topologies. In this paper we use a decomposition approach from [1], and an upper bound method from [15] to calculate and estimate the all-terminal reliability.

3 Applying an Iterated Local Search for the Communication Network Design Problem

The concept of iterated local search is a well-known metaheuristic for combinatorial optimization problems. A complete introduction to iterated local search is given in [16]. This section first introduces a new neighborhood move for the RCND problem. Afterwards an iterated local search procedure with the new move is presented.

3.1 A 1-by-2-Neighborhood Operator

To apply a local search to a problem, one has to define a move that generates a solution in the neighborhood $N(s)$. In this paper we propose the 1-by-2-move for the RCND problem. This move decreases the total cost of the network with respect to a given reliability constraint R_0. In order to generate the neighborhood $N(s)$ the move tries to delete the most costly links from the network. The complete 1-by-2-move procedure for a given configuration s represented by G_N and an edge $e_{ij} \in G_N$ is shown in Figure 1.

The procedure first tries to delete the edge e_{ij} from G_N. If the resulting graph is a valid solution, the move is accepted. If the 1-by-2-move cannot construct a valid neighbor by deleting the edge e_{ij} from G_N the procedure searches for a pair of edges that connects the vertices i and j over a vertex l using an overall cheaper pair of edges. A reliability check ensures that the replacement of the most cost-intensive edge by two overall cheaper edges represents a valid solution under the given constraint. To generate the neighborhood for a solution the 1-by-2-move is applied to all cost-intensive edges until an edge cannot be deleted or replaced. Figure 2(a) shows a sample graph. Each edge is ranked by the edge costs c_{ij}. The solid lines denote a subgraph G_N representing a solution (network)

procedure 1-by-2-move
input: e_{ij}, G_N, G , R_0
if $(R_{All}(G_N) \geq R_0$ with : $E_N \setminus \{e_{ij}\})$
  $E_N \leftarrow E_N \setminus \{e_{ij}\}$
  return G_N
candEdgesPair = $\emptyset$
for all $(\{(e_{ik}, e_{jl})|(e_{ik}, e_{jl} \in E) \wedge (e_{ik}, e_{jl} \notin E_N)\})$
  if $(k = l) \wedge (c_{ij} > (c_{ik} + c_{jl}))$
    add $\{e_{ik}, e_{jl}\}$ to candEdgesPair
sort candEdgesPair by $(c_{ik} + c_{jl})$ ascending
for $\{e_1, e_2\} \in$ candEdgesPair do
  if $(R_{All}(G_N) \geq R_0$ with: $E_N \setminus \{e_{ij}\} \wedge E_N \cup \{e_1, e_2\})$
    $E_N \leftarrow E_N \setminus \{e_{ij}\} \wedge E_N \cup \{e_1, e_2\}$
    return G_N

Fig. 1. 1-by-2-neighbor move

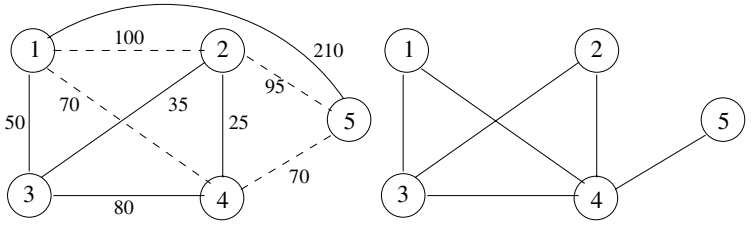


Fig. 2. Example 1-by-2-neighborhood

s_{sample} for the problem. The dashed lines in G indicate those edges currently not used in the network. For this example we use a 1-by-2-neighborhood solution for the edge e_{15}. We assume that $R_{All}(G_N) < R_0$ after the removal of e_{15}. The 1-by-2-heuristic searches two edges with total costs less than c_{15}. Candidate edges for the replacement are the edges $\{\{e_{12}, e_{25}\}, \{e_{14}, e_{45}\}\}$ (assuming that $R_{All}(G_N) > R_0$ for the candidate edges). Since the total costs of $\{e_{12}, e_{25}\}$ is 195 and the total cost of $\{e_{14}, e_{45}\}$ is 140 the 1-by-2-heuristic replaces the edge between vertex 1 and 5 with the edges $\{e_{14}, e_{45}\}$. The resulting neighbor for s_{sample} is shown in figure 2(b).

3.2 An Iterated Local Search with the 1-by-2-Move for the RCND Problem

The concept of ILS is simple and easy to implement. In order to apply ILS to a problem, one has to define an initialization method, a local search operator, a mutation operator, an acceptance criteria and a termination condition. ILS could also be used very easily for problems with a previously defined local search

procedure Iterated Local Search
input: G_N, G, R_0
s_0 = RandInitNetwork()
$s*$ = 1-by-2-LocalSearch(s_0)
do
  s' = mutate($s*$)
  s''= 1-by-2-LocalSearch(s')
  if ($C(N_{s''}) < C(N_{s*})$)
    set $s* = s''$
while ($s*$ improved in last 10 iterations)

Fig. 3. ILS procedure

operator. The steps performed using the ILS for the RNCD problem are shown in Figure 3. The ILS starts from an arbitrarily randomly-generated solution s_0. For each randomly-generated solution a reliability check to ensure that the solution is valid under the given reliability constraint. Using the 1-by-2-move the local search procedure tries to find a solution s in the neighborhood of s_0 with $C(N_s) < C(N_{s_0})$ and $R(N_s) > R_0$. The solution with the smallest $C(N_s)$ is saved as $s*$. Clearly, a solution $s*$ is a local optimal for a RCND problem.

Afterwards, the ILS procedure enters an inner loop, which iteratively starts a mutation followed by a local search using the best found solution $s*$. In order to bypass the local optima and to arrive at the global optimal solution, a mutation operator generates s' by perturbing the current best solution $s*$. The ILS mutation operator used here randomly adds currently unused edges from G to G_N. The new solution s' generated by the mutation operator is used as a starting solution for the local search procedure. With the 1-by-2-move the local search performs a local search in the neighborhood of s'. The best result found by the local search is saved in s''. A new solution s'' is accepted as a starting solution for the next ILS inner loop iteration when the total cost of the new solution $C(N_{s''})$ is less than the current best network cost $C(N_{s*})$ and the solution does not violate the reliability constraint. Otherwise the inner loop iterates with the best solution previously found. An ILS-run stops if there is no improvement for $C(N_{s*})$ in the ten previous iterations.

4 Experiments

4.1 Experimental Design

For our experiments we used a random restart local search (RRLS), an ILS and a Steady State GA (with overlapping populations). All experiments are done on a PIV- 2Ghz Linux PC. For each heuristic an initial solution is randomly generated. We use the decomposition approach from [1] and an upper bound method from [15] to calculate the all-terminal reliability. To accelerate the algorithms, the reliability calculation procedure first estimates the reliability upper bound by [15]. Only for networks with a reliability upper bound greater than R_0 the exact reliability is calculated using the method from [1]. We perform 1000 independent runs for each problem with the RRLS, and 10 independent runs for

each problem with the ILS and GA. The RRLS and LS used the 1-by-2-move from Section 3.1 to generate the neighborhood for a solution. The RRLS and ILS are implemented in C++. The GA uses the repair heuristic from [8] and is implemented in C++ using the GALib[17]. For the GA, we use a population size of 100, 50% replacement, a uniform crossover with a crossover probability of p_{cross}=0.9 and a mutation probability of p_{mut}=0.01.

4.2 Results

Table 1 summarizes the results obtained by the RRLS, the ILS and the GA. The table shows the number of nodes $|V|$ and the number of edges $|E|$, the edge reliabilities $r(l)$ and the reliability constraint R_0. The test problems are taken from [9]. Networks with the same number of nodes and same number of edges differ in the node positions and edge costs. C_{best} for the 11-nodes-problem with $r(l) = 0.9, R_0 = 0.95$ and $r(l) = 0.95, R_0 = 0.95$ are the best costs ever found. The optimal solutions C_{best} for all other test problems are published in [9]. We call the best fitness at the end of a run $C_{bestrun}$. We define D_{AVG} as the average difference (in %) for all runs between the best fitness at the end of one run and C_{best} :

$$D_{AVG} = \frac{\sum_{i=1}^{|runs|} (\frac{C_{bestrun_i} * 100}{C_{best}} - 100)}{|runs|} \tag{3}$$

where $|runs|$ is the number of runs. A high value for D_{AVG} means that there are many runs with a high difference between $C_{bestrun}$ and C_{Best}. A small value for D_{AVG} shows that an algorithm finds solutions close to C_{Best} in all runs .

D_{Best} (in %) is the difference between the best fitness of all runs and C_{best} for a test problem. Defined as:

$$D_{Best} = \frac{\min\{C_{bestrun_1} \dots C_{bestrun_{|runs|}}\} * 100}{C_{best}} - 100 \tag{4}$$

D_{best} shows the ability of an algorithm to find C_{best} in at least one run. $D_{Best} = 0$ means that the algorithm finds a solution with $C(N) = C_{best}$ in at least one run. The average number of fitness evaluations over all runs is shown by $\#Eval$.

For each problem the size of the search space is $2^{|E|}$. While a 6-nodes-test problem has only 32768 solutions the search space for a 10 nodes problem with 45 edges is already $2^{45} \approx 3.5 \cdot 10^{13}$. The results show that the RRLS is able to find optimal solutions for small problem instances (up to 7 nodes). An increase of D_{Best} for larger problems indicates that the RRLS remained in a local optimum. This can be explained by the fact that a local search method is unable to leave a local optima during a search. The higher value D_{Best} for larger problems shows that the best solutions found by the RRLS have a higher fitness difference $C_{bestrun} - C_{Best}$. This means that the RRLS often converges in a local optima. For the RRLS, the high value of D_{AVG} for all test problems shows that the RRLS only finds global optima in few of the runs, while most runs end up with a local optimal solution.

Table 1. Comparison of RRLS, ILS and GA

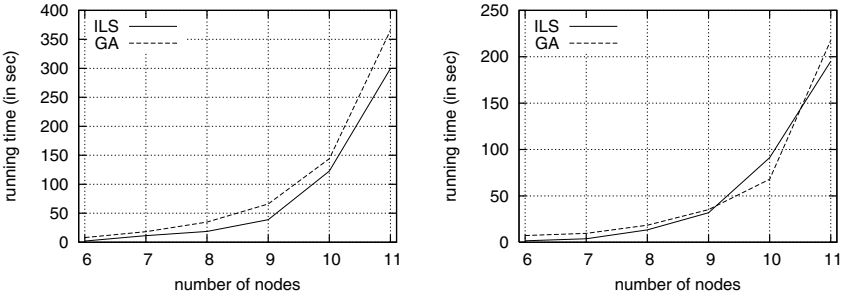
					RRLS			ILS			GA		
$\|V\|$	$\|E\|$	$r(l)$	R_0	C_{best}	D_{AVG}	D_{Best}	#Evals	D_{AVG}	D_{Best}	#Evals	D_{AVG}	D_{Best}	#Evals
6	15	0,9	0,9	231	23,33%	0,00%	10	9,52%	0,00%	49	0,00%	0,00%	288
6	15	0,9	0,9	239	41,63%	6,28%	10	6,02%	0,00%	62	0,00%	0,00%	184
6	15	0,9	0,9	227	41,35%	0,00%	10	6,17%	0,00%	55	0,00%	0,00%	171
6	15	0,9	0,9	212	50,35%	0,00%	10	2,40%	0,00%	48	0,00%	0,00%	278
6	15	0,9	0,9	184	42,13%	0,00%	10	5,00%	0,00%	44	0,00%	0,00%	193
6	15	0,95	0,95	227	20,40%	0,44%	10	13,00%	0,00%	36	1,10%	0,00%	758
6	15	0,95	0,95	213	46,63%	0,00%	10	1,31%	0,00%	46	0,00%	0,00%	284
6	15	0,95	0,95	190	60,53%	0,00%	10	13,58%	0,00%	64	0,00%	0,00%	201
6	15	0,95	0,95	200	52,29%	0,00%	10	9,50%	0,00%	33	0,50%	0,00%	436
6	15	0,95	0,95	179	41,03%	0,00%	10	6,70%	0,00%	77	0,00%	0,00%	762
7	21	0,9	0,9	189	31,72%	0,00%	14	7,67%	0,00%	124	0,00%	0,00%	598
7	21	0,9	0,9	184	57,85%	0,00%	14	1,25%	0,00%	92	0,00%	0,00%	403
7	21	0,9	0,9	243	38,03%	3,29%	14	6,42%	0,00%	94	2,18%	0,00%	418
7	21	0,9	0,9	129	53,36%	2,33%	15	4,14%	0,00%	99	0,85%	0,00%	657
7	21	0,9	0,9	124	112,22%	11,29%	15	17,10%	0,00%	90	0,00%	0,00%	319
7	21	0,95	0,95	185	30,60%	0,00%	14	7,73%	0,00%	67	0,00%	0,00%	977
7	21	0,95	0,95	182	54,57%	0,00%	14	4,51%	0,00%	75	0,00%	0,00%	782
7	21	0,95	0,95	230	40,67%	2,17%	14	4,65%	0,00%	72	1,04%	0,00%	416
7	21	0,95	0,95	122	52,44%	5,74%	14	4,84%	0,00%	79	0,57%	0,00%	846
7	21	0,95	0,95	124	104,97%	5,65%	14	13,55%	0,00%	70	0,00%	0,00%	291
8	28	0,9	0,9	208	48,49%	4,81%	17	4,71%	0,00%	149	0,00%	0,00%	401
8	28	0,9	0,9	203	55,27%	4,93%	18	0,00%	0,00%	137	0,00%	0,00%	507
8	28	0,9	0,9	211	58,97%	18,96%	14	6,93%	0,00%	84	0,00%	0,00%	685
8	28	0,9	0,9	291	42,97%	0,00%	16	2,75%	0,00%	159	0,10%	0,00%	710
8	28	0,9	0,9	178	54,02%	0,00%	19	1,01%	0,00%	143	0,84%	0,00%	887
8	28	0,95	0,95	179	59,26%	0,00%	17	3,02%	0,00%	149	0,28%	0,00%	522
8	28	0,95	0,95	194	52,40%	4,12%	18	4,28%	0,00%	128	0,31%	0,00%	836
8	28	0,95	0,95	197	46,25%	0,00%	19	5,69%	0,00%	88	0,46%	0,00%	1070
8	28	0,95	0,95	276	42,97%	0,36%	16	4,78%	0,00%	108	2,17%	0,00%	805
8	28	0,95	0,95	173	51,65%	2,31%	19	8,79%	0,00%	89	1,62%	0,00%	1133
9	36	0,9	0,9	239	57,93%	2,93%	21	6,78%	0,00%	136	0,00%	0,00%	790
9	36	0,9	0,9	191	64,26%	1,57%	23	5,85%	0,00%	134	1,57%	0,00%	979
9	36	0,9	0,9	257	42,59%	8,95%	23	6,85%	0,00%	129	2,18%	0,00%	1051
9	36	0,9	0,9	171	73,73%	4,09%	21	5,15%	0,00%	153	0,00%	0,00%	714
9	36	0,9	0,9	198	58,57%	0,51%	22	0,05%	0,00%	148	0,00%	0,00%	809
9	36	0,95	0,95	209	65,01%	0,00%	21	1,96%	0,00%	151	0,00%	0,00%	683
9	36	0,95	0,95	171	70,97%	0,00%	23	11,29%	0,00%	165	0,70%	0,00%	1261
9	36	0,95	0,95	233	48,61%	6,87%	22	8,28%	0,00%	137	0,69%	0,00%	1103
9	36	0,95	0,95	151	80,00%	0,00%	21	12,38%	0,00%	137	1,79%	0,00%	757
9	36	0,95	0,95	185	55,62%	0,00%	22	6,92%	0,00%	111	0,16%	0,00%	1062
10	45	0,9	0,9	131	42,97%	1,53%	30	4,05%	0,00%	222	0,53%	0,00%	1282
10	45	0,9	0,9	154	84,15%	10,39%	27	11,30%	0,00%	224	0,00%	0,00%	940
10	45	0,9	0,9	267	52,02%	1,87%	28	0,75%	0,00%	215	0,22%	0,00%	1207
10	45	0,9	0,9	263	45,37%	0,00%	29	2,55%	0,00%	158	0,00%	0,00%	791
10	45	0,9	0,9	293	48,08%	14,33%	26	8,58%	0,00%	194	2,87%	0,00%	1208

Table 1. (*Continued*)

					RRLS			ILS			GA		
$\|V\|$	$\|E\|$	$r(l)$	R_0	C_{best}	D_{AVG}	D_{Best}	#Evals	D_{AVG}	D_{Best}	#Evals	D_{AVG}	D_{Best}	#Evals
10	45	0,95	0,95	121	42,42%	3,31%	28	5,04%	0,00%	184	2,64%	0,00%	1614
10	45	0,95	0,95	136	88,20%	5,88%	26	15,59%	0,00%	245	0,00%	0,00%	891
10	45	0,95	0,95	236	57,28%	5,93%	27	5,44%	0,00%	198	2,29%	0,00%	1096
10	45	0,95	0,95	245	45,58%	0,00%	29	0,57%	0,00%	228	0,12%	0,00%	1037
10	45	0,95	0,95	268	49,03%	12,60%	30	10,90%	0,00%	164	0,90%	0,00%	1352
11	55	0,9	0,9	246	48,08%	9,35%	31	3,46%	0,00%	217	0,00%	0,00%	1200
11	55	0,9	0,95	277	61,35%	12,64%	34	8,45%	0,00%	230	0,00%	0,00%	1049
11	55	0,95	0,95	210	52,91%	0,48%	31	10,14%	0,00%	195	1,33%	0,00%	1543

For all problem instances the average number of fitness/reliability evaluations done by RRLS is less than that of the results obtained from the ILS and the GA. The average number of fitness evaluations for all test problems done by the RRLS is less than 35 runs. This shows the ability of the 1-by-2-move to rapidly guide the search to a local optimal solution. The results show that the extension of the LS by an additional mutation operator in an ILS method bypasses the local optima, and finally ends up with global optimal solutions. An analysis of D_{Best} for the ILS points out that the heuristic found optimal solutions for all test problems. When comparing the number of fitness/reliability evaluations (*#eval*), one finds that the ILS requires more evaluations than the RRLS, but significantly less computational effort than the GA. An analysis of the results obtained by the ILS and the GA shows that D_{Best} for the GA is equal to the D_{Best} for the ILS. But the GA performed more fitness evaluations than the ILS. The D_{AVG} measure shows that the GA, compared to the ILS, has a low diversification of the best solution in all ten runs. Compared to the D_{AVG} results for the ILS, the GA converges more often than the ILS does with the global optimal solution in all ten runs. This result can be explained by the nature of the GA heuristic. Over a GA run, solutions in the population with a low fitness quality are replaced by better solutions. At the end of a GA run only the best of the 100 solutions in the population is used for the D_{AVG} measure. In the ILS, the search process is driven by only one solution, which is not always the optimal solution of the problem in all runs.

Figure 4 shows a comparison of the running time for the ILS and GA heuristics. Each plot shows the average running time (in seconds) for all runs for the same problem class (same number of nodes) and the same configuration (link reliability and R_0). The plots in Figure 4(a) point out that the ILS is faster than the GA for all test problems. One can see that the difference between the ILS and GA running times increase as the problem size (number of nodes) grows. Figure 4(b) shows a similar result with only one exception (10 nodes). Although the GA performed more fitness evaluations than the ILS for the 10-nodes-test problems (with $r(l) = 0.95$ and $R_0 = 0.95$) the GA is faster than the ILS. This can be explained by the implemented all-terminal reliability calculation procedure which is based on a decomposition approach (see [1]). For highly reliable networks the procedure stops after a few decomposition steps and requires low



(a) $R_0 = 0.9, r(l) = 0.9$ (b) $R_0 = 0.95, r(l) = 0.95$

Fig. 4. Comparison of running time for ILS and GA

computational effort. For networks with $R_{All} \approx R_0$ the procedure must perform more decomposition steps which require a higher computational effort. This fact speeds up the GA, when compared to the ILS, as it generates many high reliable networks. The all-terminal reliability evaluation for these highly reliable networks can be done very quickly. The all-terminal reliability calculation procedure used here also explains an other interesting fact. A comparison of the running times for different configurations (link reliability and R_0) and the same problem class (same number of nodes) shows, that both heuristics run faster for $r(l) = 0.95$ and $R_0 = 0.95$ than for $r(l) = 0.9$ and $R_0 = 0.9$ although the heuristics performed more or approximately the same number of fitness evaluations. As mentioned before, this is caused by the reliability calculation procedure. If the all-terminal reliability evaluation procedure is replaced by a Monte Carlo simulation, always drawing the same number of samples for each reliability evaluation, the procedure has a constant computational effort. In this case the running time of a heuristic is proportional to the number of fitness evaluations.

5 Conclusions

This paper investigated a local search and iterated local search approach for the reliable communication network design problem. Existing approaches are capable of finding good solutions, but call for high computational effort levels. Due to the application of local search methods, the number of fitness evaluations was decreased while maintaining the same quality of solutions. We presented a new 1-by-2-move to generate the neighborhood for a solution. By dropping and replacing the most cost-intensive network links by two overall cheaper links under the given reliability constraint, the move found fitter neighbor solutions. The move connected two nodes that were previously connected directly via a third node because the indirect connection was cheaper and did not violate the reliability constraint. The empirical results showed that a local search with the 1-by-2-move often converges in a local optimum. We proposed an iterated local

search that is more efficient than existing approaches. This iterated local search with the 1-by-2-move finds global optimal solutions and outperforms a GA using a repair heuristic for a set of test problems.

References

1. Yunbin Chen, Jiandong Li, and Jiamo Chen. A new algorithm for network probabilistic connectivity. In *Military Communications Conference Proceedings*, volume 2, pages 920–923, 1999.
2. M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Fransisco, 1979.
3. Li Ying. Analysis method of survivability of probabilistic networks. *Military Communication Technology Magazine*, 48, 1993.
4. L.T.M. Berry, B.A. Murtagh, G. McMahon, S. Sudgen, and L. Welling. An integrated GA-LP approach to communication network design. *Telecommunication Systems*, 12:265–280, 1999.
5. Baoding Liu and K. Iwamura. Topological optimization model for communication network with multiple reliability goals. *Computer and Mathematics with Applications*, 39:59–69, 2000.
6. B. Dengiz and C. Alabap. A simulated annealing algorithm for design of computer communication networks. In *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics, SCI 2001*, volume 5, 2001.
7. B. Fortz and M. Labbe. A tabu search heuristic for the design of two-connected networks with bounded rings. Working Paper 98/3, Universit Catholique de Louvain, 2002.
8. Dirk Reichelt, Franz Rothlauf, and Peter Gmilkowsky. Designing reliable communication networks with a genetic algorithm using a repair heuristic. In *Proceedings 4th European Conference, EvoCOP 2004*, pages 177–187. Springer, 2004.
9. B. Dengiz, F. Altiparmak, and A. E. Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE Trans. on Evolutionary Computation*, 1(3):179–188, 1997.
10. Darren Deeter and Alice E. Smith. Economic design of reliable networks. *IIE Transactions*, 30:1161–1174, 1998.
11. Benjamin Baran and Fabian Laufer. Topological optimization of reliable networks using a-teams. In *Proceedings of World Multiconference on Systemics, Cybernetics and Informatics - SCI '99 and ISAS '99*, volume 5, 1999.
12. S. Duarte and B. Barn. Multiobjective network design optimisation using parallel evolutionary algorithms. In *Proccedings of XXVII Conferencia Latinoamericana de Informtica CLEI'2001*, Merida, Venezuela, 2001.
13. K.K. Aggarwal and Suresh Rai. Reliability evaluation in computer-communication networks. *IEEE Transactions on Reliability*, 30(1):32–35, 1981.
14. E. Manzi, M. Labbe, G. Latouche, and F. Maffioli. Fishman's sampling plan for computing network reliability. *IEEE Transactions on Reliability*, 50(1):41–46, 2001.
15. A. Konak and A. Smith. An improved general upperbound for all-terminal network reliability, 1998.
16. Helena R. Loureno, Olivier C. Martin, and Thomas Stuetzle. Iterated local search. Economics Working Papers 513, Department of Economics and Business, Universitat Pompeu Fabra, November 2000.
17. M. Wall. Galib: A c++ library of genetic algorithm components, 1998.